The fastLiquidAssociation Package

Tina Gunderson

April 16, 2015

1 Introduction

The fastLiquidAssociation package is intended to provide more efficient tools for genome-wide application of liquid association. The term liquid association (LA) was coined by K-C. Li in 2002 to describe a theory of co-expression dynamics which "conceptualize[s] the internal evolution of co-expression pattern for a pair of genes" **?**. He used the term "liquid" here (as opposed to "solid") to convey the idea of a quantity which examined the extent to which the co-expression or correlation of two genes X_1 and X_2 might be mediated by the expression of a third gene X_3, i.e. the degree to which the correlation of the two genes X_1 and X_2 might vary (e.g. go from contra- to co-expressed or vice-versa) based on the expression level of the third gene (with the third gene's expression functioning as a representation of the cellular state).

Building on Li's work, Ho et al. proposed a modification to the liquid association statistic in order to better capture more complex interdependencies between the variables, noting that, "[w]hen the conditional means and variances [of X_1 and X_2] also depend on X_3, then the three-product-moment measure, as derived by Li for standardized variables, no longer precisely estimates the expected change of co-expression of X_1 and X_2 with respect to X_3, but captures a larger set of interdependencies among the three variables" **?**.

This software package provides an expansion of the functions available in the LiquidAssociation package. While the LiquidAssociation package refers to modified liquid association (MLA) as generalized liquid association (GLA), the 2011 publication refers to it as modified liquid association. For the purposes of this vignette, we will use the two terms interchangeably. The main function (**fastMLA**) reduces the processing power and memory needed to calculate MLA values for a genome ($\binom{N}{3} * 3$, with N being the number of genes in the genome, e.g. 1.079x10^{11} for 6000 genes) by using a pre-screening method to reduce the candidate pool to triplets likely to have a high MLA value. It does this using matrix algebra to create an approximation to the direct MLA estimate for all possible pairs of X_1,$X_2|X_3$. Intuitively, we can picture MLA as the change in correlation between X_1 and X_2 when X_3 is at a high level of expression versus their correlation when X_3 is at a low level of expression. This difference, ρ_{diff} ($\rho_{diff} = \rho_{high} - \rho_{low}$), gives an approximation of the relative magnitude of the MLA value for most triplet combinations (ρ_{diff} values range from -2 to 2 compared to MLA values which range from -$\sqrt{\pi/2}$ to $\sqrt{\pi/2}$). Figure 1 shows a comparison using the first 50 and the first 250 genes of the Spellman et al. data set (available in the yeastCC package) **?** for all possible triplets of $|\widehat{MLA}|$ vs. $|\rho_{diff}|$ with lowess curves.

The package also incorporates functions to test for significance of the triplets returning a high MLA value and provides a faster, more robust version of the MLA bootstrapping estimation procedure for use with multicore systems.

2 Simple Usage

Here we present an example of package usage on a smaller data set. We start by loading the R package and the example data. In this package, we use the yeast cell cycle gene expression data by Spellman **?**. The data can be obtained through package yeastCC. The annotation package for the yeast experiment *org.Sc.sgd.db* can be obtained through Bioconductor.

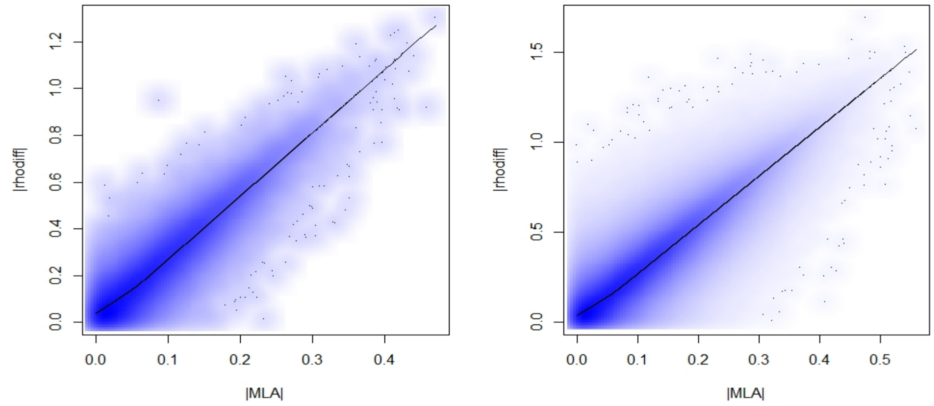

Figure 1: Comparison for all triplets possible using the first 50 and 250 genes of $|\widehat{MLA}|$ *vs.* $|\rho_{diff}|$ *with lowess curves*

2.1 fastMLA

```
==========================================================================
*
*  Package WGCNA 1.46 loaded.
*
*    Important note: It appears that your system supports multi-threading,
*    but it is not enabled within WGCNA in R.
*    To allow multi-threading within WGCNA with all available cores, use
*
*          allowWGCNAThreads()
*
*    within R. Use disableWGCNAThreads() to disable threading if necessary.
*    Alternatively, set the following environment variable on your system:
*
*          ALLOW_WGCNA_THREADS=<number_of_processors>
*
*    for example
*
*          ALLOW_WGCNA_THREADS=6
*
*    To set the environment variable in linux bash shell, type
*
*           export ALLOW_WGCNA_THREADS=6
*
*     before running R. Other operating systems or shells will
*     have a similar command to achieve the same aim.
*
==========================================================================

> library(fastLiquidAssociation)
> library(yeastCC)
> library(org.Sc.sgd.db)
> data(spYCCES)
> lae <- spYCCES[,-(1:4)]
> ### get rid of samples with high % NA elements
```

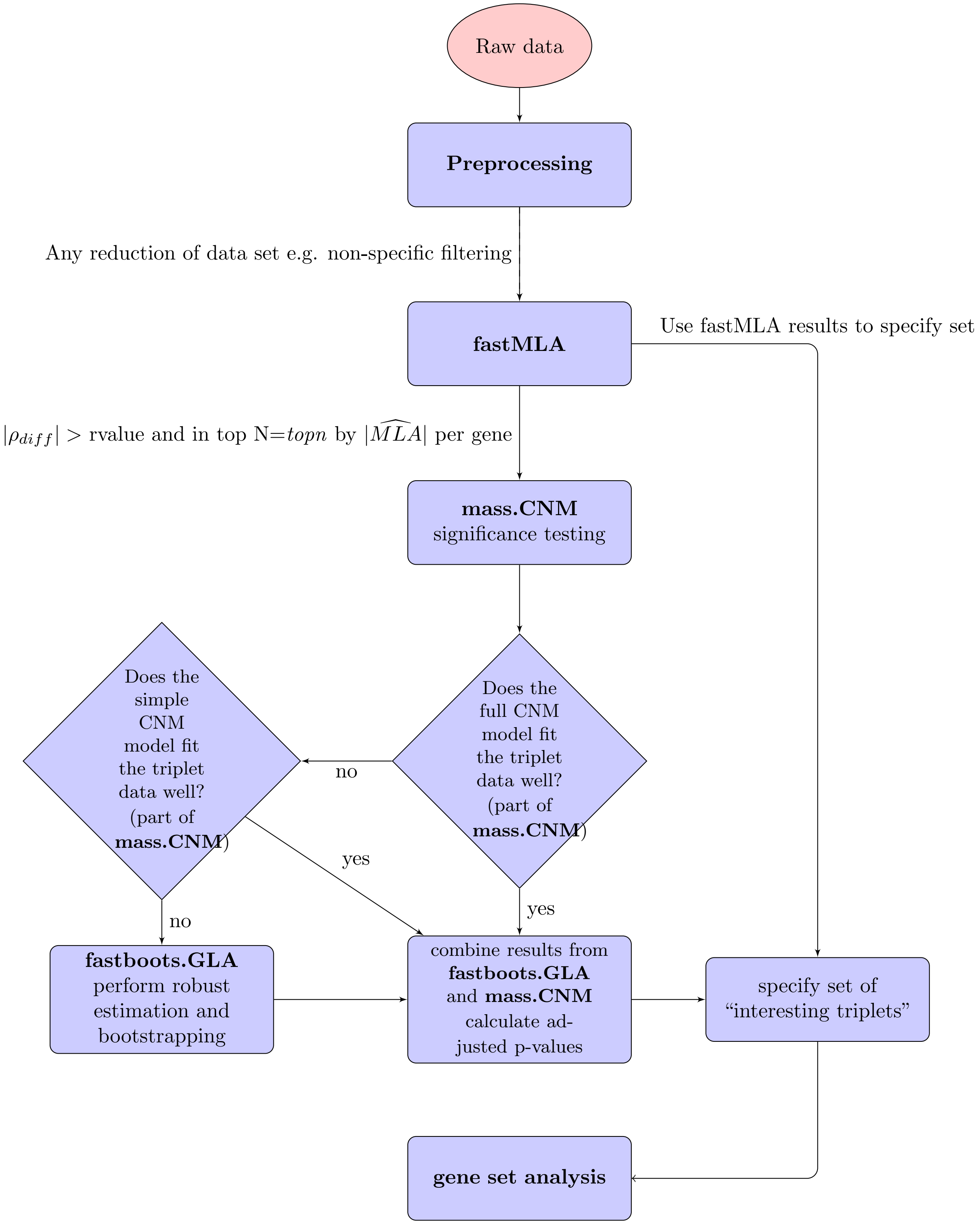
Raw data
Preprocessing
Any reduction of data set e.g. non-specific filtering
fastMLA
Use fastMLA results to specify set
$|\rho_{diff}| >$ rvalue and in top N=$topn$ by $|\widehat{MLA}|$ per gene
mass.CNM significance testing
Does the full CNM model fit the triplet data well? (part of mass.CNM)
no
Does the simple CNM model fit the triplet data well? (part of mass.CNM)
yes
yes
no
fastboots.GLA perform robust estimation and bootstrapping
combine results from fastboots.GLA and mass.CNM calculate adjusted p-values
specify set of "interesting triplets"
gene set analysis

Figure 2: Process map for data processing, testing, and gene set analysis using all components of the fastLiquidAssociation package

```
> lae <- lae[apply(is.na(exprs(lae)),1,sum) < ncol(lae)*0.3,]
> data <- t(exprs(lae))
> data <- data[,1:50]
> dim(data)
```

```
[1] 73 50
```

After removing genes with high missing percentage, for speed in this example, we reduce the data set to the first 50 genes. A normal score transformation of the data as described in both Li and Ho et al.'s work **? ?** is performed within the **fastMLA** function. The top values are returned sorted by $|\widehat{MLA}|$.

```
> detectCores()
```

```
[1] 6
```

```
> example <- fastMLA(data=data,topn=50,nvec=1:5,rvalue=0.5,cut=4,threads=detectCores())
```

```
Allowing parallel execution with up to 6 working processes.
```

```
> example[1:5,]
```

```
  X1 or X2 X2 or X1      X3  rhodiff MLA value
1  YAL041W  YAL046C YAL004W -1.14880  -0.42498
2  YAL041W  YAL042W YAL004W -1.22538  -0.40982
3  YAL014C  YAL015C YAL004W -1.09253  -0.40199
4  YAL039C  YAL045C YAL004W -1.03147  -0.39181
5  YAL025C  YAL035W YAL002W  0.94254   0.39086
```

The arguments to the **fastMLA** function are a matrix of numerical data with genes as columns(*data*), the number of results to return (*topn*), the column numbers of the genes to test (*nvec*), the $|\rho_{diff}|$ cutoff to use (*rvalue*), the *cut* argument to pass to the GLA function from the LiquidAssociation package, and the number of processors available to use for increasing the speed of calculation for correlation values (*threads*)(for a fuller discussion of this argument please see the WGCNA package documentation, for a discussion of parallelization please see the section in this vignette entitled "Parallelization of the Direct Estimate"). The *cut* argument will depend on the number of observations in the data. Based on data from Ho et al. **?**, the number of observations per bin should be in the 15-30 range for maximum specificity. The *cut* argument is equal to the number of bins plus one. Hence in this example, as we have 73 observations, we chose to se the *cut* value at 4. The maximum *rvalue* is theoretically 2 (as $\rho_{diff} = |\rho_{high} - \rho_{low}|$ and -1 $\leq \rho_{X_1,X_2} \leq 1$). The default value is set at 0.5 or 25% of the realizable correlation difference, however $|\rho_{diff}|$ a.k.a. *rvalue* was set at 1.1 in our example above for speed purposes. Caution should be used when determining the *rvalue*. Too high a value risks missing those triplets whose MLA values are not fully reflected by the more simplistic correlation, while too low a value approaches testing all possible triplets and forfeits any increase in testing efficiency. [1]

2.2 Functions of the Conditional Normal Model

Two options for calculating significance are included in this package, **mass.CNM** which relies on estimation of parameters from a conditional normal model and **fastboots.GLA** which uses a more robust direct estimate method. Both functions return unadjusted p-values. The CNM model is written as:

$$\begin{array}{rcl} X_3 & \sim & N(\mu_3, \sigma_3^2) \\ X_1, X_2|X_3 & \sim & N(\left(\begin{array}{c} \mu_1 \\ \mu_2 \end{array} \right), \Sigma). \end{array}$$

[1] The fastMLA function incorporates the more efficient correlation calculation of the **cor** function in the WGCNA package **?**.

where $\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$. The mean vector (μ_1, μ_2) and variance matrix Σ depend on the level of X_3 as written below:

$$\begin{aligned} \mu_1 &= \beta_1 X_3, \\ \mu_2 &= \beta_2 X_3, \\ \log \sigma_1^2 &= \alpha_3 + \beta_3 X_3, \\ \log \sigma_2^2 &= \alpha_4 + \beta_4 X_3, \\ \log\,[\frac{1+\rho}{1-\rho}] &= \alpha_5 + \beta_5 X_3. \end{aligned}$$

```
> #from our example with fastMLA
> CNMcalc <- mass.CNM(data=data,GLA.mat=example,nback=5)
> CNMcalc

$`top p-values`
  X1 or X2 X2 or X1      X3 rhodiff MLA value estimates san.se  wald p value
1  YAL011W  YAL017W YAL005C    0.79      0.30      0.70   0.17 17.14       0
2  YAL040C  YAL048C YAL005C    0.87      0.31      0.67   0.20 10.87       0
3  YAL031C  YAL036C YAL002W   -0.78     -0.30     -0.79   0.24 10.42       0
4  YAL028W  YAL041W YAL004W    0.94      0.30      1.22   0.38 10.03       0
5  YAL041W  YAL046C YAL004W   -1.15     -0.42     -1.55   0.50  9.51       0
  model
1     F
2     F
3     F
4     F
5     F

$`bootstrap triplets`
NULL
```

In our original subset of the data, there are no triplets not returning sensible values, so to demonstrate what is returned when sensible values are not returned for either model, we change the dataset used to the full 5721 genes in the *yeastCC* data set and use a matrix which specifically contains triplets not returning sensible values.

```
> fulldata <- t(exprs(lae))
> load(system.file('data','testmat.RData',package='fastLiquidAssociation'))
> notsense <- testmat
> CNMother <- mass.CNM(data=fulldata,GLA.mat=notsense,nback=5)
> CNMother

$`top p-values`
  X1 or X2 X2 or X1      X3 rhodiff MLA value estimates san.se    wald p value
1  YOL120C  YOL143C YDR495C  1.2710    0.5376    1.5562 0.4030 14.9116  0.0001
2  YHR129C  YPR045C YDR495C  1.4467    0.5474    2.4887 0.6495 14.6822  0.0001
3  YIR043C  YLR375W YDR495C  1.3415    0.5358    1.7832 0.5269 11.4531  0.0007
4  YIR004W  YOR291W YDR495C -1.4788   -0.5662   -2.1373 0.6774  9.9541  0.0016
5  YCRX15W  YDR257C YDR495C -1.3919   -0.5435   -2.1025 0.6693  9.8679  0.0017
  model
1     F
2     F
3     F
4     F
```

```
5     F

$`bootstrap triplets`
     X1 or X2  X2 or X1  X3        rhodiff  GLA value estimates
[1,] "YKR032W" "YNR021W" "YDR294C" "1.2808" "0.4567"  "22.1364"
[2,] "YBR012C" "YDL191W" "YDR346C" "1.4023" "0.4298"  "-10211.647"
     san.se          wald     p value  model
[1,] "20.0702"       "1.2165" "0.27"   "S"
[2,] "15288623.7058" "0"      "0.9995" "S"
```

In the above example, we apply the **mass.CNM** function to the results of **fastMLA** to test for significance using both of the conditional normal model-based approaches laid out in Ho et al. **?**. The **mass.CNM** function accepts three arguments: *data*, *GLA.mat*, and *nback*. *data* is the same matrix used as *data* in the **fastMLA** function, the *GLA.mat* is the returned matrix from a call to the **fastMLA** function, and *nback* is the number of results to return, sorted by p-value, then by size of the Wald statistic. The main parameter of interest in the CNM for significance testing using liquid association is b_5.

The **mass.CNM** incorporates a two step process, first testing the triplets against the full CNM model and then for any triplets which do not return sensible values (sensible being defined as any triplet for which the p-value or SE estimate=0, SE estimate >10, or any β_5 statistics returned "NaN", "NA", or "Inf"), it tests them against the simple model. The full CNM model uses all parameter value (i.e. estimates the parameters of the model above using all five equations), while the simple model only uses the last three.

The **mass.CNM** output is a list with two components, one a data frame with *nback* rows which list the information from fastMLA (the triplet, the triplet's GLA and ρ_{diff} values), the β_5 values estimated in the model (the estimate for β_5, its SE, the Wald statistic, and p-value), and the model (full or simple) which was used to calculate the values and the second a list of any triplets who did not return sensible values for either model. For any triplets that do not return a sensible value using one of the two models, the more robust direct estimate method is available as the **fastboots.GLA** function. [2]

2.3 Parallelization of the Direct Estimate

As mentioned before, for triplets that when tested with the full or simple models did not return sensible values, a direct estimator is available. The **fastboots.GLA** function makes use of the increased processing power that has become more widely available, both on standard desktops and servers. It is an updated and parallelized version of the getsGLA function available in the LiquidAssociation package. As the standard error in the direct estimate relies on bootstrapping and the significance calculations can require a large number of iterations, we sought a more efficient method to increase efficiency.

Parallelization has as its basis the idea that large problems (e.g. a high number of calculations) can be split into smaller problems that are being run at the same time, rather than sequentially. Parallelization of processes makes sense when each process itself takes significantly longer to complete than to allocate.
In the below example, we use the **detectCores** function from the parallel package to detect the number of CPU cores. On systems where more processors are available, e.g. servers for supercomputing which normally have multiple nodes of several processors each, it is possible to set a larger number of cores than the **detectCores** function may detect.

fastboots.GLA is intended to be used with the results of a call to **mass.CNM**, specifically the output labelled bootstrap triplets. The **fastboots.GLA** function is intended for use on multicore systems in order to make use of possible parallelization. While it can be used on a single core system, there would be no speed benefit over the **getsGLA** function available in the LiquidAssociation package, though the **fastboots.GLA** function does resolve an error in the **getsGLA** function that can be caused by data redundancy. In the below example, we are using a six core system.

```
> #determine number of processors for multicore systems
> cores <- detectCores()
> cores
```

[2] The above example may produce a warning message "In sqrt(diag(object$valpha)) : NaNs produced". This is normal behavior when one or both of the CNM models cannot fit at least one of the triplets.

```
[1] 6
```

```
> clust <- makeCluster(cores)
> boottrips <- CNMother[[2]]
> dim(boottrips)
```

```
[1]  2 10
```

Below we an example of processing speed with the fastboots.GLA function. Using the getsGLA function from the LiquidAssociation package, the triplet had the following values: [3]

user	system	elapsed	sGLA	p value
30.85	0.01	30.92	5.25	0.00000

```
> #We take the results for the single triplet and put it in matrix format
> example.boots <- t(as.matrix(boottrips[1,]))
> dim(example.boots)
```

```
[1]  1 10
```

```
> set.seed(1)
> system.time(GLAnew <- fastboots.GLA(tripmat=example.boots,data=fulldata,
+ clust=clust, boots=30, perm=500, cut=4))
```

```
   user  system elapsed 
  2.732   0.000  18.512 
```

```
> GLAnew
```

```
  X1 or X2 X2 or X1      X3 rhodiff MLA value MLA stat boots p-value
1  YKR032W  YNR021W YDR294C  1.2808    0.4567  5.25085             0
```

```
> #the matrix conversion is not needed for the 2 line result
> set.seed(1)
> system.time(GLAtwo <- fastboots.GLA(tripmat=boottrips,data=fulldata,
+ clust=clust, boots=30, perm=500, cut=4))
```

```
   user  system elapsed 
  2.538   0.001  24.298 
```

```
> GLAtwo
```

```
  X1 or X2 X2 or X1      X3 rhodiff MLA value MLA stat boots p-value
1  YKR032W  YNR021W YDR294C  1.2808    0.4567 5.250850    0.00000000
2  YBR012C  YDL191W YDR346C  1.4023    0.4298 2.748675    0.02419355
```

```
> stopCluster(clust)
```

tripmat here is the list of triplets where a direct estimate is needed. The *data* is the numeric matrix of data used in both the **fastMLA** and **mass.CNM** functions. *clust* is a cluster of CPU cores to use in parallelization, created by a call to the **makeCluster** function from package parallel. The *boots* argument specifies the number of bootstrap iterations for estimating the bootstrap standard error. The *perm* argument specifies the number of iterations to calculate the permuted p-value. Given that the results of the permutations depend on random number generation, for reproduciblity here we use the set.seed option. The *cut* option is used to determine the number of "buckets" to used in the estimation procedure.

Because the normalization of the data matrix is incorporated into the **fastboots.GLA** function, the full time decrease is not apparent until the number of iterations significantly exceeds the time to normalize the data matrix or the number of triplets to test increases. This is partially displayed in the example, where even the calculation of two triplets has a lower processing time than the single triplet run serially.

[3] The original getsGLA code is not called here as it sometimes produces an error due to non-unique cut values with higher numbers of permutations.

3 Further Applications

In this section, we cover further applications available with the results of the fastLiquidAssociation package. Using the GOstats package in conjunction with the results from either the **fastMLA**, the **mass.CNM**, or **fastboots.GLA** we can specify our set of interesting genes and perform GO analysis. We show an example here using the genes in the X_3 position of the results from **fastMLA** to define our list of interesting genes and a p-value cutoff of 0.05 for the BP ontology. Only categories containing a minimum of five genes are displayed.

```
> library(GOstats)
> library("org.Sc.sgd.db")
> ##X3 genes
> topX3 <- unique(example[,3])
> hyp.cutoff <- 0.05
> ####
> params <- new("GOHyperGParams", geneIds=topX3,universeGeneIds=colnames(data),
+ annotation="org.Sc.sgd.db",ontology="BP",pvalueCutoff=hyp.cutoff,conditional=TRUE,
+ testDirection="over")
> GOout <- hyperGTest(params)
> summary(GOout,categorySize=5)
```

```
      GOBPID     Pvalue OddsRatio  ExpCount Count Size
1 GO:0070727 0.01187172  23.40000 0.6666667     3    8
2 GO:0008104 0.02451434  15.85714 0.8333333     3   10
3 GO:1902580 0.04864323  13.66667 0.4166667     2    5
                                   Term
1   cellular macromolecule localization
2                  protein localization
3 single-organism cellular localization
```

Because we used open reading frame (ORF) IDs in order to use the GOstats package with yeast data, we can the following to unpack the associated gene names.

```
> ###extracts GO list elements of summary(hyperGtestobj)<cutoff
> ###converts ORFids to Gene names and returns under GO list element
> ##for ontology BP
> GOids <- summary(GOout,categorySize=5)$GOBPID
> check <- GOout@goDag@nodeData@data
> subset <- check[GOids]
> terms <- summary(GOout,categorySize=5)$Term
> test <- sapply(subset,function(m) m[1])
> orflist <- lapply(test,function(x) intersect(x,topX3))
> ##creates mapping of ORFids to gene names
> x <- org.Sc.sgdGENENAME
> mappedgenes <- mappedkeys(x)
> xx <- as.list(x[mappedgenes])
> mapid <- names(xx)
> ##creates list of GO ids
> genename1 <- lapply(orflist,function(x) xx[match(x,mapid)])
> ###
> ###if reduced num of terms desired use next line else skip to next ##
> for(i in 1:length(genename1)){
+ if (length(genename1[[i]])>10) genename1[[i]]<-genename1[[i]][1:10]
+ }
> ##for full list use below
> genelist <- lapply(genename1,function(x) paste(x,collapse=", "))
> ugenes <- unlist(genelist)
> names <- sapply(names(ugenes), function(x) unlist(strsplit(x, split='.', fixed=TRUE))[1])
> umat <- matrix(ugenes)
> umat <- cbind(terms,umat)
> rownames(umat) <- names
> colnames(umat) <- c("GO description","Associated genes")
> umat

           GO description                             Associated genes
GO:0070727 "cellular macromolecule localization"   "TFC3, VPS8, SSA1"
GO:0008104 "protein localization"                  "TFC3, VPS8, SSA1"
GO:1902580 "single-organism cellular localization" "VPS8, SSA1"
```

4 References

5 Session Information

```
R version 3.2.0 (2015-04-16)
Platform: x86_64-unknown-linux-gnu (64-bit)
Running under: Ubuntu 14.04.2 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] grid      stats4    parallel  stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
 [1] GOstats_2.34.0              graph_1.46.0
 [3] Category_2.34.0             GO.db_3.1.2
 [5] Matrix_1.2-0                fastLiquidAssociation_1.4.0
 [7] Hmisc_3.15-0                ggplot2_1.0.1
 [9] Formula_1.2-1               survival_2.38-1
[11] lattice_0.20-31             LiquidAssociation_1.22.0
[13] org.Sc.sgd.db_3.1.2         AnnotationDbi_1.30.0
[15] GenomeInfoDb_1.4.0          IRanges_2.2.0
[17] S4Vectors_0.6.0             yeastCC_1.7.1
[19] Biobase_2.28.0              BiocGenerics_0.14.0
[21] geepack_1.2-0               WGCNA_1.46
[23] RSQLite_1.0.0               DBI_0.3.1
[25] fastcluster_1.1.16          dynamicTreeCut_1.62

loaded via a namespace (and not attached):
 [1] genefilter_1.50.0     reshape2_1.4.1        splines_3.2.0
 [4] colorspace_1.2-6      XML_3.98-1.1          RBGL_1.44.0
 [7] foreign_0.8-63        RColorBrewer_1.1-2    matrixStats_0.14.0
[10] foreach_1.4.2         plyr_1.8.1            stringr_0.6.2
```

```
[13] munsell_0.4.2          gtable_0.1.2           codetools_0.2-11
[16] latticeExtra_0.6-26    doParallel_1.0.8       preprocessCore_1.30.0
[19] GSEABase_1.30.0        proto_0.3-10           Rcpp_0.11.5
[22] acepack_1.3-3.3        xtable_1.7-4           scales_0.2.4
[25] annotate_1.46.0        impute_1.42.0          digest_0.6.8
[28] tools_3.2.0            cluster_2.0.1          MASS_7.3-40
[31] reshape_0.8.5          iterators_1.0.7        AnnotationForge_1.10.0
[34] rpart_4.1-9            nnet_7.3-9
```